Periodica Polytechnica
Electrical Engineering and Computer Science

The Effect of Latency Increasing on the Realisation Cost in High Level Synthesis of Pipeline Systems

58(2), pp. 37-42 (2014)
DOI: 10.3311/PPee.7024

György Pilászy / *György* Rácz / *Péter* Arató

RESEARCH ARTICLE

RECEIVED 27 AUGUST 2013; ACCEPTED AFTER REVISION 25 FEBRUARY 2014

Abstract

This paper examines the effects of increasing the latency in pipeline systems. The high level synthesis methods focus on the pipeline throughput only, and the latency is an output parameter. The proposed method is capable for reducing the cost of the implementation by increasing the latency at the same throughput. The essence of the proposed method is that an increase in the latency may increase the mobility ranges of the processing units. Thus, the increased degrees of freedom may cause better implementation by affecting the scheduling and allocation steps. An impact assessment algorithm for calculating proper latency increment range is also presented.

Keywords

Pipelining · latency time · HLS · CAD · microcontroller · multiprocessing · embedded systems

György Pilászy
Department of Control Engineering and Information Technology,
Faculty of Electrical Engineering and Informatics, BME,
Magyar tudósok krt 2., H-1117 Budapest, Hungary
e-mail: gpilaszy@iit.bme.hu

György Rácz
Department of Control Engineering and Information Technology,
Faculty of Electrical Engineering and Informatics, BME,
Magyar tudósok krt 2., H-1117 Budapest, Hungary
e-mail: gyuriracz@iit.bme.hu

Péter Arató
Department of Control Engineering and Information Technology,
Faculty of Electrical Engineering and Informatics, BME,
Magyar tudósok krt 2., H-1117 Budapest, Hungary
e-mail: arato@iit.bme.hu

1 Introduction

During the high-level synthesis of the pipeline systems, the most important parameter is the throughput [1, 3, 6]. The effect of increasing the latency period is not analysed. However, a longer latency does not slow down the system in pipeline sense, since the effect is negligible for the throughput.

One of the aims of this paper is to determine the effect of increasing the latency to the cost function in the pipeline systems. For demonstration, PIPE HLS tool (developed by BME-IIT) has been used in the analysis, but the results are not affected by using commercial tools (e.g. Catapult C). The cost reduction is expected, since the extent of overlapping between the elementary operations may be reduced by increasing the latency. Thus, the scheduling and allocation steps of HLS may result in a less expensive structure by taking advantage of better reusing situation. An illustration is shown in Figure 1. The processing of the first data is marked in (red) dark, and of the second data in (blue) bright. The unidirectional arrows represent the data dependence. For simplicity, let the $e_1 \ldots e_5$ elementary operations be identical. The processing time of each elementary operation is assumed 8 clock cycles. The elementary operations e_2 and e_3 start and finish at the same time. Thus they can not be implemented in the same processing unit. The overlapping will be eliminated, for example, if e_2 starts at a later time (t_3). This case is shown in Figure 2. It can be observed that the latency has increased, because of the execution time of e_2 and the data dependency.

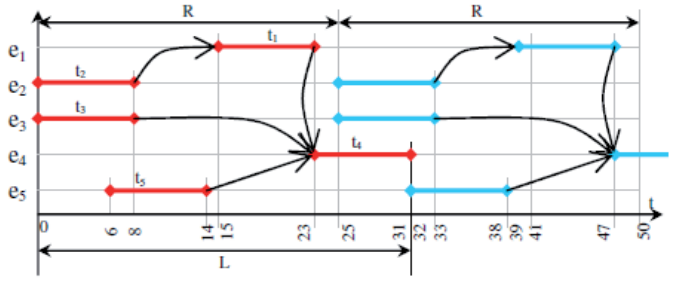

Fig. 1. Scheduling of a pipeline system (***L=31, R=25***)

Since one processing unit can perform at most three elementary operations within the specified restart time ($3 \cdot 8 < R = 25$), therefore at least two processing units should be used. This situation is shown in Figure 2. The elementary operations e_1, e_2, e_3, and e_4, e_5 can be allocated into two processing units, respectively. In consequence, the latency is increased by 1 ($L = 32$).

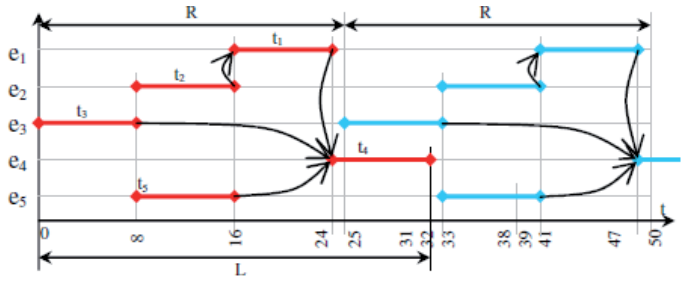

Fig. 2. Scheduling with increased latency (L=32, R=25)

A further reduction of the required number of processing units may be achieved by applying longer restart time. An extreme situation is illustrated in Figure 3. This solution requires only one processing unit, because there are no overlapped elementary operations, but there is no pipelining any more ($R = L$).

The organization of the remaining sections is as follows. In section 2, a simple cost function will be assumed for illustrating the effect of the longer latency, and then simulation results are presented as case studies. In section 3, an algorithm is described for calculating the increased latency. The paper ends with the conclusion in section 4.

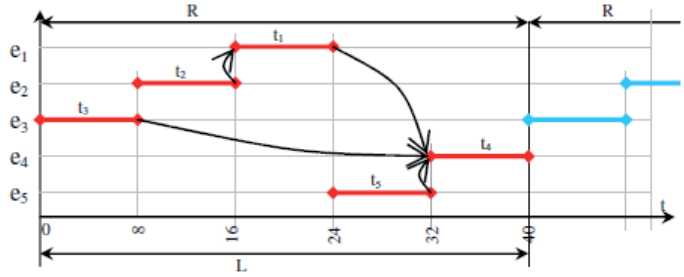

Fig. 3. Scheduling with increased restart time and latency (L=40, R=40)

2 Case studies

In this section, we present examples to illustrate the cost reduction effect of the increased latency. Firstly, we define our assumed simple cost function.

2.1 The simple cost function

In our example, each node of the input graph contains one elementary operation. Let t_i denote the execution or processing time of the elementary operation e_i. After the scheduling and allocation steps, more elementary operations can be allocated into one processing unit (P_i). The aim is to minimize the number of these processing units. Let C_i denote the cost of processing unit P_i. If the number of processing unit P_i is n_i

$$C_i = n_i \cdot t_i$$

The simple total cost (C) is the sum of partial costs C_i:

$$C = \sum_{\forall i} C_i = \sum_{\forall i} n_i \cdot t_i$$

By calculating this cost for the examples of Figures 1-3, the results are shown in Table 1.

Tab. 1. The costs calculated for Figures 1..3

R	L	n_i	C
25	31	3	24
25	32	2	16
40	40	1	8

There are two ways for cost reduction by increasing the latency. The simplest way is to increase the latency step-by-step by one, to execute the whole HLS software each time and to check the new total cost value. The disadvantage of this method is the great number of re-run steps that takes too much time. The other way is to determine only those latency increasing steps in advance, which might decrease the total cost. Some rules can be formulated for finding these favourable latency increasing steps as follows.

If the same types of elementary operations start simultaneously, then the latency should be increased at least by the value of the execution time in order to allocate them into the same processing unit. For reusing a processing unit (P_i) k times within a restart period (R), it is necessary that the overall execution time of elementary operations allocated into P_i must not be greater than the restart time(R):

$$\sum_k t_k \le R,$$

If the execution times of all elementary operations allocated in P_i are identical, then
$k \cdot t_k \le R$ must hold.

Further on, the above procedure is illustrated by a simple benchmark from [1].

2.2 FFT algorithm

In the following example [1], there is an eight-point FFT (Fast Fourier Transformation) algorithm, the elementary operation graph of which is shown in Figure 4. In Table 2, the assumed execution times are summarized.

Tab. 2. Elementary operations of the FFT

Execution units	Operation	t_i	pieces
P1	MUL	12	12
P2	SUB	3	12
P3	ADD	3	12

The initial value of the latency is $L = 48$. This is calculated by PIPE from the original graph for $R = 10$ restart time. Since the shortest elementary operation time (t_i) is 3, the restart period should be at least $R>6$ in order to achieve a beneficial effect by increasing L.

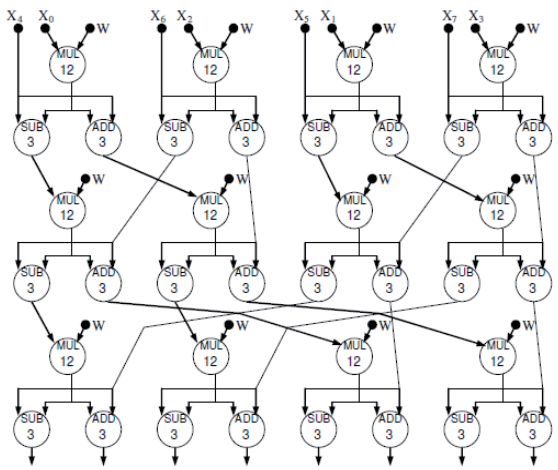

Fig. 4. Elementary operation graph of the FFT algorithm

The time range of scanning the restart period: is chosen for $R = \{12,...,49\}$

The range of the latency value: $L = \{48, 51, 54, 60, 96\}$

Table 3 shows the cost functions based on Table 2. The cost functions are shown in Figure 5. The procedure has been carried out also in case of multifunction execution units assumed to be able to execute both SUM and ADD elementary operations. Table 4 shows the number of required multifunction execution units. The cost functions are shown in Figure 6.

It can be observed, that the cost values are not always smaller at the longest latency. The reason of this is the approximate character of the scheduling algorithm actually applied in the HLS tool. Namely, the start times of the elementary operations are fixed somehow depending on each other, and so sometimes stricter constraints may arise for the allocation algorithm. In the above examples, the force-directed scheduler of the HLS tool PIPE was applied.

2.3 Observations:

1. Increasing the latency value increases the mobility of the elementary operations. The larger mobility increases the running time of the scheduling and allocating algorithm because of the higher degrees of freedom. However, this higher degree of freedom may give better results.
2. If $R \leq 2 \cdot min\{t_i\}$, then no significant decrease of the cost can be expected, because of the replication and so with frequent restarting, none of the execution units can be reused.
3. If an e_i elementary operation has the execution time $t_i \geq R$, then no significant improvement can be expected in the number of the units executing e_i, because of the replication performed by the HLS tool [1].

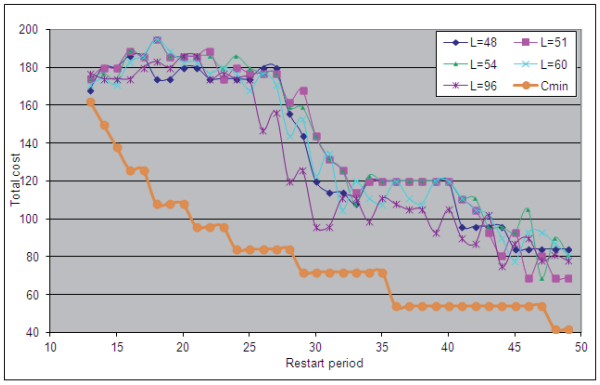

Fig. 5. Cost function of the FFT algorithm

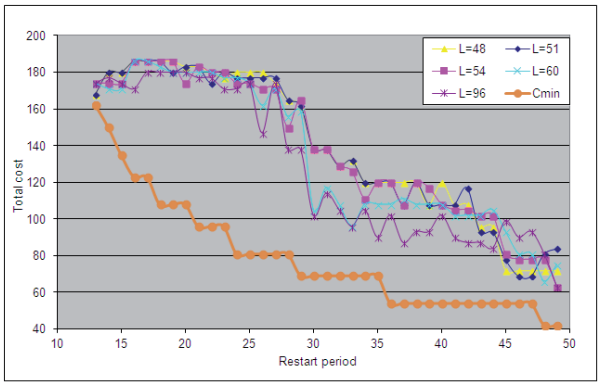

Fig. 6. Cost function of the FFT algorithm in case of multifunction execution units

3 The proposed algorithm

Let the following notations be introduced:

L_{min}: the latency value determined by the HLS tool for the desired restart time R

L_{max}: an arbitrary value (greater than L_{min}) defined by the user, typical value is $2 \cdot L_{min}$

e_j: an elementary operation (a node) in EOG, the range of j consists of the integers from 1 until the number of nodes in EOG

$E\{e_1, \ldots e_j, \ldots e_m\}$: The set of the elementary operations in EOG.

$\Phi_F\{f_1, f_2, \ldots, f_k, \ldots f_n\}$: A complete cover of the elementary operations, where block f_k contains all elementary operations with type k. (It is cover, because one e_j can be executed by more than one execution unit.)

t_k: the execution time of the elementary operations belonging to block f_k. (this value is identical for all operations belonging to the same block).

n_k: the number of the elementary operations in block f_k

P_i: the execution unit type i defining an incomplete cover on Φ_F denoted by Φ_i, Te blocks of Φ_i contain those elementary operations from the blocks of Φ_F, which are executable by P_i

M_q: the q-th copy of an execution unit after the allocation (its type is determined by a P_i

$M\{M_1, \ldots, M_q, \ldots M_z\}$: the set of the necessary copies of the execution units determined by the allocation.

$\Pi_M \{m_1, \ldots, m_i, \ldots m_p\}$: the complete partition of the execution unit copies. (Block m_i contains all those copies M_q, the type of which is determined by P_i.

N_i: the number of copies in bolck m_i, i.e. the number of neccessary copies of P_i.

T_i: the longest execution time of elementary operations which are executable by P_i.

C_i: The cost of applying execution unit P_i determined by the allocation: $C_i = N_i \cdot T_i$

C: the total cost: $C = \sum_i C_i$

C_d: a minimal cost value, defined by the user for stop.

N_{imin}: the minimal number of P_i copies for a given restart time R. This value of R could not be achieved by applying less copies of P_i

C_{min}: the minimal cost $C_{\min} = \sum_i \mathrm{N}_{\mathrm{imin}} \cdot T_i$

3.1 The main steps of the algorithm:

1. Determine the minimal value of the latency time (L_{min}) by the HLS tool. [1]
2. Determine to each P_i belong the minimal number of the execution units (rounded up).

$$N_{iMIN} = \sum_k \left\lceil \frac{n_k \cdot t_k}{R} \right\rceil$$

3. Determine the minimal cost (C_{min}). In case of fixed restart time (R), the lowest possible cost can be estimated by the following equation:

$$C_{\min} = \sum_{\forall i} N_{iMIN} \cdot T_i$$

If the value of R can be increased, then the theoretical minimum cost is obtained when each elementary operation occur as few times as possible: $n_1 = n_2 = \ldots n_i = 1$. Then the minimum cost value:

$$C_{\min} = \sum_{\forall i} T_i$$

This result represents already a non pipeline situation, because the elementary operations do not overlap each other: $R \geq L$.

4. Create sets $M\{M_1, M_q, \ldots, M_z\}$ of execution units after the allocation. To each M_q belongs the execution time t_i representing the longest execution time of the elementary operations executed by the execution units in M_i.
5. Create $\Pi_M \{m_1, \ldots, m_i, \ldots, m_p\}$ full partition of M according to the P_i then select those, where $|m_i| > N_{iMIN}$ and $t_i \leq R/2$
6. Select the smallest t_i for incrementing L according to the $max(|m_i|)$
7. Increase the latency by the selected Ti and rerun the HLS tool. (go to step 4.)
8. The new latency $L+t_i$ yields a new cost value C
9. There might be set a limit L_{max} of incrementing the latency. A possible default value could be: $L_{max} = 2 \cdot L_{min}$.
   The stopping condition can be formulated in many ways. One possibility is that we stop the iteration when we reached the predefined maximum latency value (L_{max}). This condition can be tested easily.
   Another possibility is that the iteration is continued until the theoretically achievable minimum cost (C_{min}) or a desired cost (C_d) is reached.

The combination of both methods can give good results. In each iteration step, it should be examined which of the above conditions is achieved (L_{max} or C_{min}). If any of these conditions are fulfilled, the iteration would be stopped.

4 Conclusion

The proposed method can reduce the cost of hardware implementation of a task by increasing the latency time in pipeline systems while maintaining the same throughput. The method helps in designating efficient ranges for incrementing the latency, and enables the impact assessment of a range regarding the total cost.

Acknowledgement

The support of the Hungarian Scientific Fund (OTKA K72611) and New Hungary Development Plan (Project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0002) IKT-P5-T3 are gratefully acknowledged. This work is also belonging to the scientific program of the "Development of quality-oriented and harmonized R+D+I strategy and functional model at BME" project.

References

1 **Arató P., Visegrády T., Jankovits I.,** *High Level Synthesis of Pipelined Datapaths*. New York: John Wiley & Sons, (2001).

2 **Pilászy Gy., Móczár G.,** *Remote control of modular microcontroller systems, Microcad 2001*. In: International Scientific Conference Section F: Measurement and Automation. Miskolc, 1th March 2001-2th March 2001, pp. 45-50, (2001).

3 **Arató P., Drexler D., Kocza G., Suba G.,** *Synthesis of a Task-dependent Pipelined Multiprocessing Structure*. ACM Transactions on Design Automation of Electronic Systems. (submitted to)

4 **Coussy P., Gajski D. D., Meredith M., Takach A.,** *An Introduction to High-Level Synthesis*. IEEE Design & Test of Computers, 26 (4), pp. 8-17, (2009).
DOI: 10.1109/MDT.2009.69

5 **Pilászy Gy., Rácz Gy., Arató P.,** *Communication Time Estimation in High Level Synthesis*. Periodica Polytechnica Electrical Engineering and Computer Science, 57 (4), pp. 99-103, (2013).
DOI: 10.3311/PPee.7413

6 **Fingeroff M.** *High-Level Synthesis Blue Book*. Xlibris Corporation, (2010).

7 **Xu J., Wolf W., Henkel J., Chakradhar S.,** *A design methodology for application-specific networks-on-chip*. ACM Transactions on Embedded Computing Systems, 5 (2), pp. 263-280, (2006).
DOI: 10.1145/1151074.1151076

8 **Mann Z. Á., Orbán A., Arató, P.,** *Finding optimal hardware/software partitions*. Formal Methods in System Design, 31 (3), pp. 241–263, (2007).
DOI: 10.1007/s10703-007-0039-0

9 **Arató P., Mann Z. Á., Orbán A.,** *Algorithmic aspects of hardware/software partitioning*. ACM Transactions on Design Automation of Electronic Systems, 10 (1), pp. 136–156, (2005).
DOI: 10.1145/1044111.1044119

10 **Suba G.,** *Hierarchical pipelining of nested loops in high-level synthesis*. submitted to Periodica Polytechnica Electrical Engineering and Computer Science, 58, (2014).
DOI: 10.3311/PPee.7610

Tab. 3. Total cost of execution units of the FFT algorithm

R	L=48	L=51	L=54	L=60	L=96	C_{min}
13	168	174	174	171	177	162
14	180	180	177	174	174	150
15	180	180	174	171	174	138
16	186	189	189	183	174	126
17	186	186	186	186	180	126
18	174	195	195	195	183	108
19	174	186	186	189	180	108
20	180	186	186	183	186	108
21	180	186	186	183	186	96
22	174	189	186	177	174	96
23	174	174	180	180	177	96
24	174	180	186	177	174	84
25	174	177	180	168	174	84
26	180	177	177	177	147	84
27	180	177	177	171	156	84
28	156	162	159	144	120	84
29	144	168	159	153	126	72
30	120	144	144	123	96	72
31	114	132	132	135	96	72
32	114	126	126	105	111	72
33	108	114	108	120	111	72
34	120	120	123	111	99	72
35	120	120	120	108	111	72
36	120	120	120	120	108	54
37	120	120	120	111	105	54
38	120	120	120	108	105	54
39	120	120	120	120	93	54
40	120	120	120	120	105	54
41	96	111	111	111	90	54
42	96	105	111	105	87	54
43	96	93	96	102	102	54
44	96	81	96	90	75	54
45	84	93	93	78	87	54
46	84	69	105	93	90	54
47	84	81	69	93	78	54
48	84	69	90	87	81	42
49	84	69	81	81	78	42

Tab. 4. Total cost of multifunction execution units of the FFT algorithm

R	L=48	L=51	L=54	L=60	L=96	C_{min}
13	168	168	174	174	174	162
14	180	180	174	171	177	150
15	180	180	174	171	174	135
16	186	186	186	186	171	123
17	186	186	186	186	180	123
18	186	186	186	183	180	108
19	186	180	186	180	180	108
20	183	183	174	180	180	108
21	183	183	183	180	177	96
22	180	174	180	180	177	96
23	177	180	180	177	171	96
24	180	177	174	177	171	81
25	180	177	174	174	174	81
26	180	177	171	162	147	81
27	174	177	171	171	174	81
28	165	165	150	156	138	81
29	162	162	165	159	138	69
30	138	138	138	105	102	69
31	138	138	138	117	114	69
32	129	129	129	108	105	69
33	132	132	126	96	96	69
34	120	120	111	108	105	69
35	120	120	120	108	90	69
36	120	120	120	108	102	54
37	120	108	108	111	87	54
38	120	120	120	108	93	54
39	108	108	117	108	93	54
40	120	108	108	108	102	54
41	108	108	105	102	90	54
42	108	117	105	102	87	54
43	96	93	102	102	87	54
44	96	93	102	105	84	54
45	72	78	81	93	99	54
46	72	69	78	81	90	54
47	72	69	78	81	93	54
48	72	81	78	66	81	42
49	72	84	63	75	63	42